FSD Laboratory 01

Name - Shounak Dighe
Roll No - 20
Pannel - H
Prn - 1032233107

Aim: Version control with Git.

Objectives:

1. To introduce the concepts and software behind version control, using the example of Git.
2. To understand the use of 'version control' in the context of a coding project.
3. To learn Git version control with Clone, commit to, and push, pull from a git repository.

Theory

1. What is Git? What is Version Control?

Version Control:

- **Definition:** Version control is a system that records changes to files over time so that you can recall specific versions later. It helps track changes, collaborate with others, and manage different versions of a project.
- **Types:** There are two main types of version control systems:
    - **Local Version Control Systems:** Track changes on a single machine.
    - **Distributed Version Control Systems (DVCS):** Track changes on multiple machines (e.g., Git).

Git:

- **Definition:** Git is a distributed version control system that allows multiple people to work on a project simultaneously. It keeps track of changes in your source code and helps coordinate work among programmers.
- **Key Features:**
    - **Branching and Merging:** Git allows you to create branches for different features or fixes and merge them back into the main codebase.
    - **Distributed Nature:** Every user has a full copy of the repository, including its history.
    - **Efficiency:** Git is designed to handle large projects efficiently.

2. How to Use Git for Version Controlling?

Here's a basic workflow to use Git for version control:

1. **Clone a Repository:**
    - Use the `git clone` command to create a local copy of a remote repository.
    - Command: `git clone [repository URL]`
2. **Make Changes and Commit:**

- **Stage Changes:** Use `git add [file]` to stage files for commit.
- **Commit Changes:** Use `git commit -m "commit message"` to save your changes with a message describing them.

3. **Push Changes to Remote Repository:**
   - Use `git push` to upload your local commits to the remote repository.

FAQ

1. What is Branching in Git?

Branching:

- **Definition:** Branching allows you to diverge from the main line of development and continue to work independently. It's useful for developing features, fixing bugs, or experimenting with new ideas without affecting the main codebase.
- **Common Branches:**
  - **Main Branch (main or master):** The primary branch where the stable code resides.
  - **Feature Branches:** Created for developing specific features or fixes.

2. How to Create and Merge Branches in Git?

Creating a Branch:

- Command: `git branch [branch-name]`
- This creates a new branch but does not switch to it. To create and switch to a new branch in one step, use:
  - Command: `git checkout -b [branch-name]`

Switching Branches:

- Command: `git checkout [branch-name]`

Merging Branches:

- First, switch to the branch you want to merge into (typically `main`):
  - Command: `git checkout main`
- Then, merge the branch into the current branch:
  - Command: `git merge [branch-name]`

Screenshots

While I can't create screenshots directly, here's how you can take them yourself:

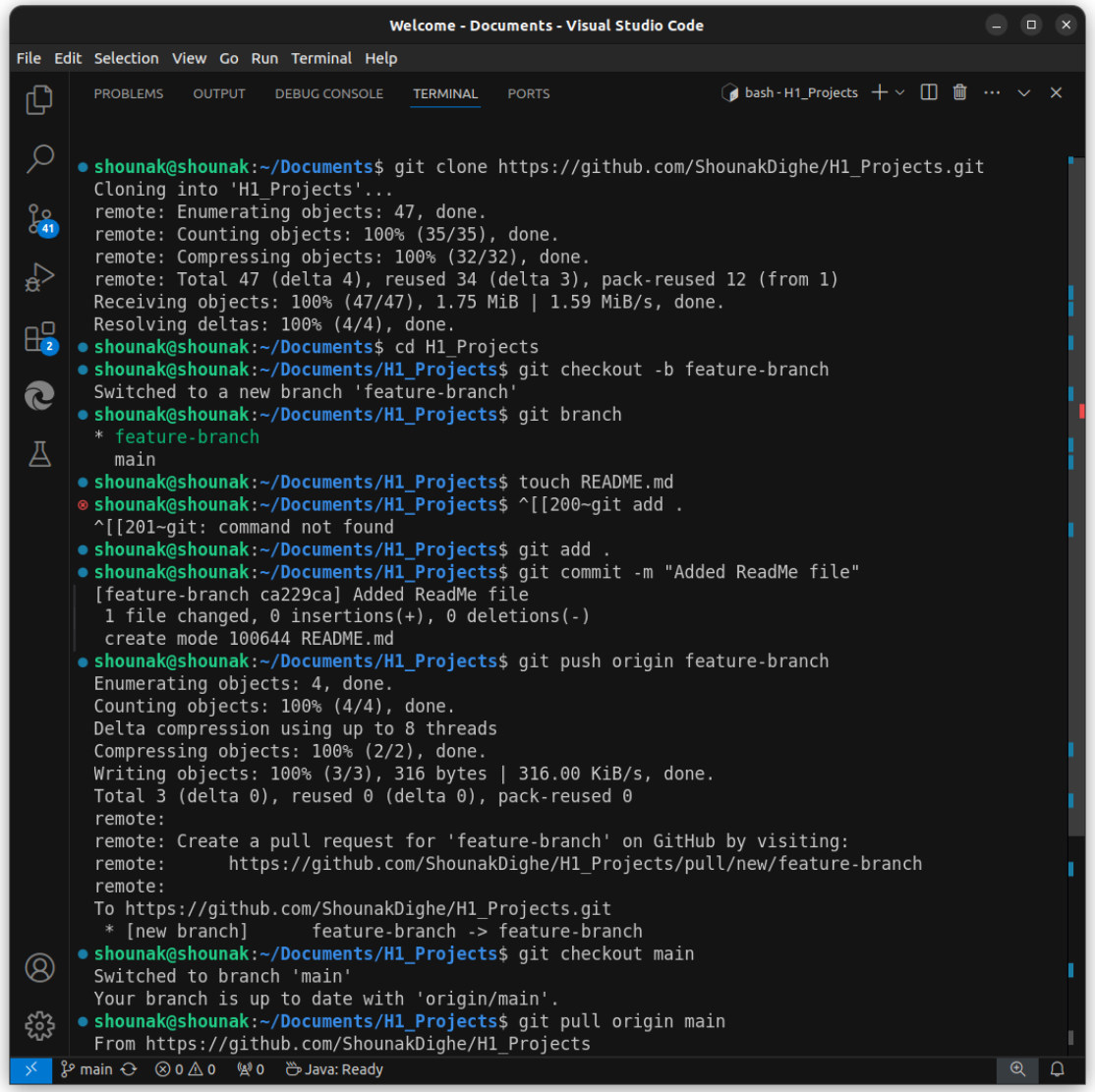
1. **Cloning a Repository:**

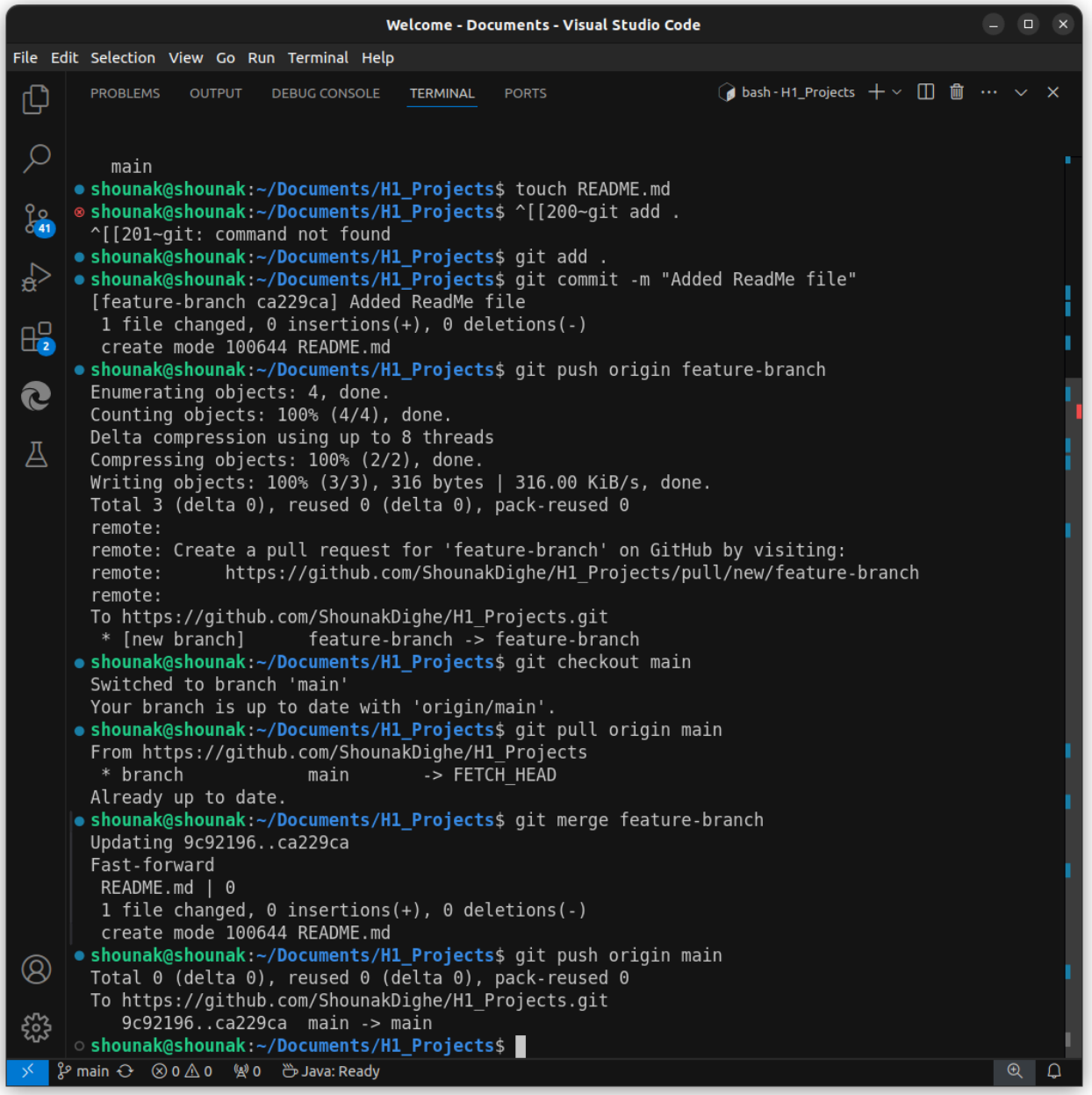
- Run `git clone [repository URL]` in your terminal and take a screenshot of the output.

2. **Making Changes and Committing:**
   - After making changes, run `git status`, `git add [file]`, and `git commit -m "message"`. Capture the terminal output of each command.
3. **Pushing and Pulling Changes:**
   - Run `git push` and `git pull`, and take screenshots of the terminal output.
4. **Branching Commands:**
   - Use `git branch`, `git checkout -b [branch-name]`, and `git merge [branch-name]`. Capture the results of each command in your terminal.

Problem Statement:

Create a public git repository for your team and submit the repo URL as a solution to this assignment, Learn Git concept of Local and Remote Repository, Push, Pull, Merge and Branch.





The README.md file is push to git repository